Hierarchically Partitioned Implicit Surfaces For Interpolating Large Point Set Models

David T. Chen[1], Bryan S. Morse[2], Bradley C. Lowekamp[1], and Terry S. Yoo[1]

[1] National Library of Medicine, Bethesda, MD, USA
[2] Brigham Young University, Provo, UT, USA

Abstract. We present a novel hierarchical spatial partitioning method for creating interpolating implicit surfaces using compactly supported radial basis functions (RBFs) from scattered surface data. From this hierarchy of functions we can create a range of models from coarse to fine, where a coarse model approximates and a fine model interpolates. Furthermore, our method elegantly handles irregularly sampled data and hole filling because of its multiresolutional approach. Like related methods, we combine neighboring patches without surface discontinuities by overlapping their embedding functions. However, unlike partition-of-unity approaches we do not require an additional explicit blending function to combine patches. Rather, we take advantage of the compact extent of the basis functions to directly solve for each patch's embedding function in a way that does not cause error in neighboring patches. Avoiding overlap error is accomplished by adding phantom constraints to each patch at locations where a neighboring patch has regular constraints within the area of overlap (the function's radius of support). Phantom constraints are also used to ensure the correct results between different levels of the hierarchy. This approach leads to efficient evaluation because we can combine the relevant embedding functions at each point through simple summation. We demonstrate our method on the Thai statue from the Stanford 3D Scanning Repository. Using hierarchical compactly supported RBFs we interpolate all 5 million vertices of the model.

1 Introduction

Many applications in computer graphics require smoothly interpolating a large set of points on or near a surface. These points may originate as unorganized point sets such as from a 3-D scanning system. They may also come in organized or semiorganized sets from the vertices of polygonal models, which once interpolated can provide a smoother surface than the polygonal one and can be converted to other representations, including a more finely polygonalized one if desired. Such point sets may also come from computer vision analysis of an image volume or from interactive modeling tools.

Numerous techniques have emerged for converting point sets to implicit models that interpolate (or approximate) these points [1–15]. Broadly, we call these *interpolating implicit surfaces*.[3] These methods take the same general approach: known points on the surface define where the implicit surface's embedding function should have a value of 0; known off-surface points, surface normals (either known or fitted), or other assumptions

[3] These have also been known in the literature as *variational implicit surfaces*, *implicit surfaces that interpolate*, and *constraint-based implicit surfaces* by various authors.

define where the embedding function has nonzero values; and the embedding function is then interpolated using scattered data interpolation techniques such as radial basis functions (RBFs) [1, 2, 4–7, 10], (implicit) moving least squares [14], or partition-of-unity blending of local fitting using these or other interpolation methods [8, 11, 12].

Many implementations [1, 2, 6] use thin-plate splines, but this requires the solving of a large, full, generally ill-conditioned system of equations and quickly becomes computationally impractical for large models. Other RBFs with infinite support [7, for example] have similar limitations. Various methods have been used to accelerate RBF approaches, including using compactly supported RBFs to make the required system sparse [4, 9, 10, 13, 16] or approximating a large set of constraints by a well-selected subset [3, 5]. Others accelerate the surface fitting by subdividing the surface into smaller patches, fitting a surface (or the embedding function for that surface) to each patch, then combining the patches through blending [8, 11, 12, 14].

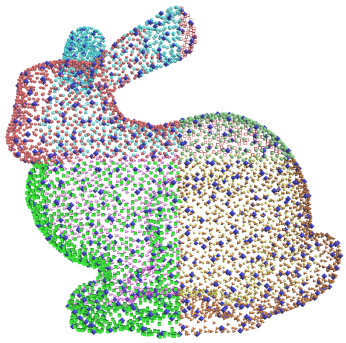


Fig. 1. A simple two-level hierarchy of the Stanford bunny model. The constraints of the root RBF are dark, larger octohedra, and each child partition is indicated in different shapes, styles and colors.

This paper presents a novel method for efficiently creating compactly supported RBF-based implicit representations from the vertices of a polygonal model using hierarchical spatial partitioning as illustrated in Figure 1. Unlike other approaches that combine local interpolations through compactly supported blending functions, no explicit blending function is required—each level and partitioned patch is calculated so that a simple linear combination of them produces an exact interpolation.

2 Related Work and Background

Our implementation of compactly-supported RBFs follows most closely that detailed in [4], which is based on the general approach of Turk *et al.* [2, 6].

The basic method begins with a set of points known to lie on the desired implicit surface and constrains the interpolated embedding function to have a value of 0 at these points. Using the method of [6], non-zero constraints (often called "normal constraints") are placed at fixed offsets in the direction of the known or desired normals at these points. This produces a set of point/value constraints $P = \{(\mathbf{c}_i, h_i)\}$ such that $h_i = 0$ for all $\mathbf{c}_i$ on the surface and $h_i = 1$ for all $\mathbf{c}_i$ at a fixed offset from that surface. An embedding function $f(\mathbf{x})$ is then interpolated from these constraints such that $f(\mathbf{c}_i) = h_i$.

This interpolation is done using an RBF $\phi(r)$ by defining the embedding function f as a weighted sum of these basis functions centered at each of the constraints:

$$f(\mathbf{x}) = \sum_{(\mathbf{c}_i, h_i) \in P} d_i \, \phi(\|\mathbf{x} - \mathbf{c}_i\|) \tag{1}$$

where d_i is the weight of the radial basis function positioned at $\mathbf{c}_i$.[4] To solve for the unknown weights d_i, substitute each constraint $f(\mathbf{c}_i) = h_i$: into Eq. 1:

$$\forall \mathbf{c}_i : f(\mathbf{c}_i) = \sum_{(\mathbf{c}_j, h_j) \in P} d_j \phi(\|\mathbf{c}_i - \mathbf{c}_j\|) = h_i \tag{2}$$

By using compactly supported RBFs one can make this system of equations sparse [4, 9]. By efficiently organizing the points spatially, one can also reduce the time required to compute the system itself. As the size of the model increases, one can commensurately reduce the radius of support for the RBFs, thus increasing efficiency while keeping the data density approximately constant.

The primary drawback to using compactly supported radial basis functions alone for surface modeling is that the embedding function is 0 outside one radius of support from the surface. This does not preclude polygonalization, ray-tracing, or many other uses of the surface because it is relatively easy to separate zero sets that result from lack of support. However, it does limit their use for CSG and other operations for which implicit surfaces are useful. The compact support also causes them to fail in areas with low data density, in the limit failing where the surface has holes larger than the support. (See [10] for an excellent discussion of the limitations of compactly supported RBFs for surface modeling, with additional empirical analysis in [16].) These limitations can be overcome using hierarchical, or multilevel, approaches, such as [17, 18] for scattered data interpolation, and [10, 13] for compactly supported RBFs.

Another way to accelerate the surface fitting is to spatially subdivide the surface points into separate patches, then interpolate (or approximate) each patch and blend the results using partition-of-unity or similar blending techniques [8, 11, 12, 14]. By blending local approximations instead of directly trying to fit the entire model at once, this method provides efficient processing for very large models. Key to these partition-of-unity or other blending methods is the use of a compactly supported weighting function to blend separate patches or the effects of individual points in a neighborhood.

We demonstrate a new method for creating and blending interpolations for separate patches that uses compactly supported RBFs to interpolate the patches and, due to their compactly supported nature, does not require the a separate explicit blending function.

3 Method

As with other hierarchical partitioning approaches, our method builds a large-scale approximating embedding function then successively refines it with smaller-scale incremental functions. The two main components of our method are selecting the points for each node in the hierarchy and creating *phantom constraints* to handle overlapping function domains. Using phantom constraints to clamp each embedding function allows us to combine them simply by addition rather than requiring a blending function.

3.1 Building a Hierarchy

To build a hierarchy we use an octree to span the input points, which is traversed from the top down. Points are first selected for the root, producing an embedding function for

[4] For some RBFs, including thin-plate splines, an additional polynomial may be required.

a base model. Then points for the each of the children of the root are selected, adding detail at a finer resolution. After solving the refining embedding functions for the eight children, we proceed to the grandchildren, and so on, stopping when all points have been included in the hierarchy. When building any given node of the hierarchy, the functions for the nodes above it have already been solved.

Selecting Points For a Node Points in a node's octant are selected based on a random Poisson-disk distribution. However, our initial implementation, the traditional Poisson-disk distribution where there is a minimum Euclidean distance between any two points, tended to undersample regions of high curvature. We needed to allow sample points to be closer together in high curvature regions.

Comparing the normal directions of nearby points is an efficient estimate of local curvature. A region with points that have disparate normals requires a higher sampling rate. To achieve adaptive sampling we use a modified distance function based on the points' normals. If two points have identical normals, the distance between them is the same as the Euclidean distance. However, if their normals differ we would like them to appear to be farther apart. The net effect is to place samples closer together in areas of higher curvature.

We have experimentally developed an admittedly arbitrary modified distance function that scales the Euclidean distance between points $\mathbf{x}_1$ and $\mathbf{x}_2$ by a quadratic function of the angle θ between the normal vectors:

$$\text{dist}(\mathbf{x}_1, \mathbf{x}_2, \theta) = \|\mathbf{x}_1 - \mathbf{x}_2\| \left(\tfrac{1}{2} \cos(\theta)^2 - \tfrac{7}{2} \cos(\theta) + 4 \right) \tag{3}$$

Selecting Points For the Root Selecting points for the root embedding function is especially important because error in the root propagates throughout the hierarchy. The more error there is in a parent node, the more "energy" required at a child node to bend the embedding function to fit. Since the root node affects all other nodes, we are more particular in selecting its points.

At the root node, in addition to the modified Poisson-disk distribution mentioned earlier, we attempt to select points that are "representative" of a local region. The goal is to pick points that capture the larger-scale shape of a region, pushing smaller scale detail or noise to nodes lower in the hierarchy.

For the root node, candidate points are screened by comparing the normal direction for each point with the normals of points around it and rejecting those that are too disparate. Specifically, if the average dot product of a candidate point's normal with the normals around it is below a specific threshold (0.1), the point is not selected.

Figure 1 shows a two-level hierarchy of RBFs of the Stanford bunny. The hierarchy consists of a root RBF and eight children. The nodes' constraints are differentiated by style. The root's constraints are dark blue, slightly larger octahedra.

Embedding Function For a Node Once points have been selected for interpolation within a node, interior, surface, and exterior constraints are placed for each point. The embedding function also requires a level set value for each constraint and a radius of

support for the compact RBF. The root embedding function should produce the correct results at the locations of the root's constraints. Therefore the constraint values at the root are determined solely by the type of constraint. By default the functions' values at surface, interior and exterior constraints should be 0, 1 and -1 respectively.

Using the notation of Eqs. 1 and 2, we can write the root embedding function f_0 defined by the set of root constraints $P_0 = \{(\mathbf{c}_i, h_i)\}$ using root-level RBF $\phi_0(r)$:

$$f_0(\mathbf{x}) = \sum_{(\mathbf{c}_i,h_i)\in P_0} d_{0i}\ \phi_0(\|\mathbf{x}-\mathbf{c}_i\|) \tag{4}$$

where the root-level weights d_{0i} are determined by solving the system of equations

$$\forall \mathbf{c}_i \in P_0 : f_0(\mathbf{c}_i) = h_i \tag{5}$$

The embedding function of a child node is an increment that corrects the parent function at the location of the child node's constraints. For example, at a child node's surface constraint the net function should to evaluate to 0. However, the parent function evaluates to some value α. Therefore the child's function should evaluate to $-\alpha$ to correct the error. Thus at each child constraint location, the hierarchy of embedding functions above the child node is evaluated, and a value is given to the child constraint that corrects the result of the nodes above it.

Thus, we may write a single child level's embedding function f_1 defined by the child node's constraints $P_1 = \{(\mathbf{c}_i, h_i)\}$ and child-level RBF $\phi_1(r)$, along with the parent node's constraints P_0 and embedding function f_0, as follows:

$$f_1(\mathbf{x}) = \sum_{(\mathbf{c}_i,h_i)\in P_0\cup P_1} d_{1i}\ \phi_1(\|\mathbf{x}-\mathbf{c}_i\|) \tag{6}$$

where the child-level weights d_{1i} are determined by solving the system of equations

$$\forall \mathbf{c}_i \in P_0 \cup P_1 : f_1(\mathbf{c}_i) = h_i - f_0(\mathbf{c}_i) \tag{7}$$

Note that each level k of the hierarchy uses its own RBF $\phi_k(r)$ and weights d_{ki}.

Since Eq. 5 already holds for the root constraints, the root embedding function f_0 already evaluates correctly at the root constraints and no correction is required by the child embedding function:

$$\forall \mathbf{c}_i \in P_0 : f_1(\mathbf{c}_i) = 0 \tag{8}$$

This process may be continued to additional levels of the hierarchy and extended to include multiple nodes at each level. Solving for and combining embedding functions for multiple nodes at each level is addressed in Section 3.2.

Once all the constraint values for a node have been determined, the radius of support for the compact RBF for that node must be determined. We attempt to keep the same number of points per node, and nodes at different levels in the hierarchy cover different-sized regions. Naturally different-level nodes should have compact RBFs with different radii. In our approach the user selects the radius for the root, and each descendant is given a radius proportional to that root radius and to its own size.

Typically compact RBFs are used for the embedding functions of all of the nodes in the hierarchy, but using only compactly supported RBFs have the problem of the

function being undefined in some regions. Any location that is outside of all constraints' radii of support will not have a defined embedding function. Therefore we also allow the option of using a thin plate spline (TPS) RBF for the root node (see Figure 3), eliminating the problem. Although a TPS is much more expensive to solve and evaluate, the embedding function for the root node uses only a limited subset of points, making it still practical even for otherwise large models.

3.2 Phantom Constraints

Managing embedding functions with overlapping extent is a common problem that occurs when attempting to partition a point set. To interpolate all the points in a data set, we must guarantee that the combination of all embedding functions that impinge on a point produces the exact value we require. Our task is simplified by the compact RBF's limited extent. Therefore at any given point only relatively few embedding functions need to be combined and evaluated.

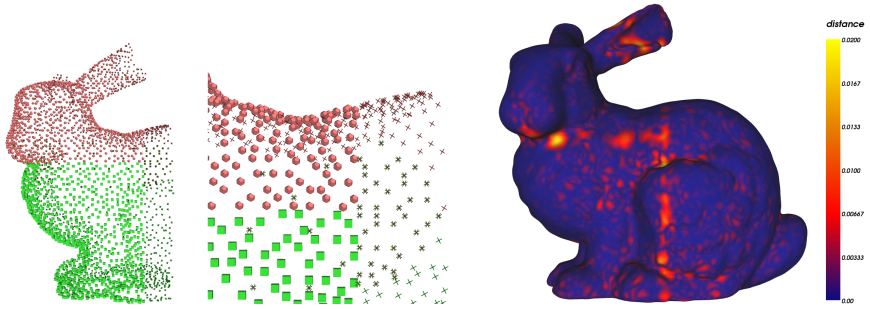
Our approach is to place *phantom constraints* in a given node to clamp its embedding function. Phantom constraints are placed in regions where the extent of the embedding functions for different nodes overlap, requiring us to suppress the influence of the node's embedding function. In this way, phantom constraints serve much the same purpose as the blending function in partition-of-unity approaches but without explicit blending during evaluation of the implicit surface's embedding function. The locations for phantom constraints fall into two categories: locations that have been inherited from regular constraints in ancestral nodes, and locations from regular constraints in adjacent sibling nodes. Using a top-down approach means constraint locations from descendant nodes can be ignored.

A child node's embedding function is an incremental change applied to the sum of its ancestor embedding functions as mentioned in Section 3.1. For our purposes an ancestor node is any node in the octree above a given node whose embedding function overlaps with that node, not just direct ancestors. Since a child embedding function is an increment to the functions above it, a regular constraint of the child is given a value that corrects the summed ancestor functions. Also, phantom constraints with values of 0 are placed in the child RBF at all ancestor constraint locations within the child's bounds to ensure the child's function does not produce erroneous results at these locations.

Similarly, the interpolation within a node should not be incorrectly affected by neighboring sibling nodes. Therefore, any regular constraints of neighboring siblings that overlap with the extent of a node's embedding function become corresponding phantom constraints for the node.

Extending the notation of Eqs. 4–8, we define $P_{lk} = \{(\mathbf{c}_i, h_i)\}$ as the set of constraints for node k of level l. We also define $\hat{P}_{lk} = \{(\mathbf{c}_i, h_i)\}$ as the set of phantom constraints relevant to this node and the function $\hat{f}_{lk}$ as the sum of all other embedding functions relevant to this node (i.e., those higher up in the hierarchy whose support-expanded regions overlap this node's support-expanded region). We may thus write the embedding function for this child node in terms of these constraints and the node's RBF $\phi_{lk}(r)$ as

$$f_{lk}(\mathbf{x}) = \sum_{(\mathbf{c}_i, h_i) \in P_{lk} \cup \hat{P}_{lk}} d_{lki}\ \phi_{lk}(\|\mathbf{x} - \mathbf{c}_i\|) \tag{9}$$



a b

Fig. 2. Phantom constraints. a) the constraints of two neighboring child nodes of the bunny. The solid shapes are regular constraints, and the crosses are phantom constraints, rotated based on node. b) the effects of phantom constraints on the embedding function. The left side of the bunny does not have phantom constraints from neighboring nodes, while the right side does. The color shows the distance error between the embedding function and the original surface.

where the child node's weights d_{lki} are determined by solving the system of equations

$$\forall \mathbf{c}_i \in P_{lk} \cup \hat{P}_{lk} : f_{lk}(\mathbf{c}_i) = h_i - \hat{f}_{lk}(\mathbf{c}_i) \tag{10}$$

Again, this node's embedding function provides incremental refinement only and does not change the result at the phantom constraints from other nodes:

$$\forall \mathbf{c}_i \in \hat{P}_{lk} : f_{lk}(\mathbf{c}_i) = 0 \tag{11}$$

Figure 2a shows two overlapping child nodes of the bunny. The solid shapes represent regular constraints, while the crosses represent phantom constraints. The phantom constraints contained with the bounds of a node's octant have been inherited from the root node, while those outside the octant come from neighboring sibling nodes.

Figure 2b illustrates the error that can occur from overlapping embedding functions that do not have phantom constraints. The left side of the bunny has no phantom constraints from neighboring octants, while the right side does have phantom constraints. The surface is colored by the distance error between the embedding function and the original surface mesh. Clearly there is much more error on the left side, particularly where octants abut. Also the error bleeds into the right side of the bunny because the functions on the left are not evaluating to 0 to the right.

Figure 3 shows slices through four embedding function of the bunny. Images 3a and 3b use a compact RBF for the root node, while 3c and 3d use a thin plate spline at the root. In the left pair the textured region is where the compact RBF is not defined. Images 3a and 3c are slices through the embedding functions of just the root nodes. The images that slice through two level hierarchies (3b and 3d) clearly show sharper boundaries and more detail. For instance, the bottoms of the bunny are less rounded.

Adding phantom constraints outside of a given node's bounds expands the region of space where the node's RBF must be evaluated. However, this expansion can be nullified by only defining the embedding function in the original region defined by the regular constraints. At any location with only phantom constraints within the RBF's radius, the embedding function returns 0.

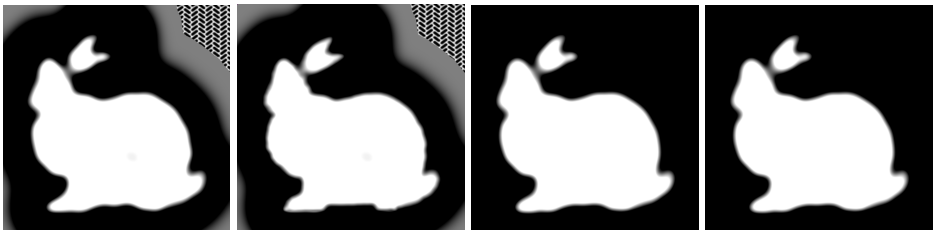


Fig. 3. Slices through the embedding functions. a) a compact RBF root. b) a two level hierarchy, with a compact root. c) a thin plate spline root. d) a two level hierarchy, with a thin plate spline root. The texture regions are where the compact RBF is undefined.

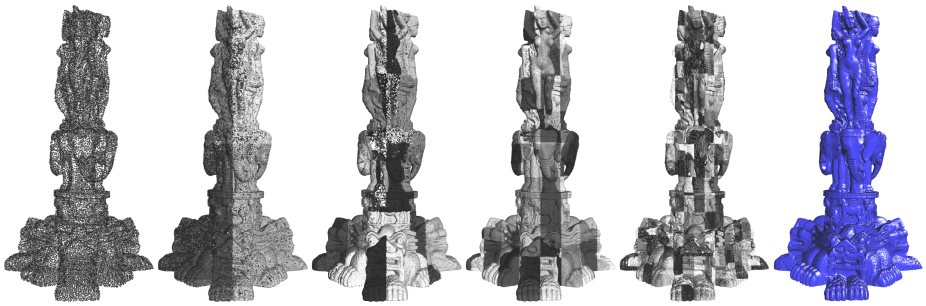


Fig. 4. Stanford's Thai statue. The 5 levels of the hierarchy of constraints and an extracted iso-surface. The hierarchy contains 17.5 million constraints.

4 Results

To demonstrate the efficacy of our method, we applied it to the Thai statue from Stanford's 3D Scanning Repository. The model consists of 5 million vertices to which we added 426,245 vertices on the bottom, which was not scanned. Our implicit model has 17.5 million constraints in a 5 level octree. Figure 4 shows the five levels in the hierarchy and an iso-surface extracted from the embedding function.

Section 4.1 analyzes the statistics and error characteristics of our method. Section 4.2 provides some implementation details.

4.1 Statistics

Figure 5 shows statistics for the implicit hierarchy. The upper section of the table shows statistics for the entire model, and the middle section shows statistical averages per implicit evaluation. The bottom section shows the error in the implicit surface. To compute the statistics in the middle and bottom sections, the embedding function was evaluated at every surface constraint location in the data set.

The implicit error of a constraint is the unsigned difference between the value a constraint should have and the value returned by the embedding function. By default, surface constraints should have a value of 0.

The distance error is the distance between a surface constraint's location and a root (zero) of the embedding function. The root was found by searching the embedding function along the constraint's normal direction. The data set has an extent 395.9 along its longest axis, and the exterior constraints were offset by a distance of 10^{-4}. Values of 0 for surface constraints and -1 for exterior constraints of the embedding function results in gradients on the order of 10^4. Thus one would expect implicit errors on the order of 10^4 times greater than the distance errors, as the statistics demonstrate.

		Thai statue
Hierarchy statistics	*vertices*	4,999,996
	regular constraints	10,852,316
	phantom constraints	6,632,801
	total constraints	17,485,117
	tree nodes	799
	tree height	5
	build time (minutes)	127.35
Averages per evaluation	*regular constraints*	961.77
	phantom constraints	202.00
	number of nodes	6.59
Error	*avg. implicit error*	1.4980E-06
	max. implicit error	1.4950E-04
	avg. distance error	2.1230E-10
	max. distance error	2.2700E-08

Fig. 5. Statistics for the hierarchical model of the Thai statue.

In our examples, adding phantom constraints increases the number of constraints in the data sets by an average of 61.1% so that phantom constraints make up 37.9% of the constraints. However, since the phantom constraints tend to occur towards the bottom of the hierarchy, i.e. in the nodes with smaller extents, the average number of phantom constraints encountered per function evaluation is lower. In all they represent only 17.4% of the constraints when evaluating the embedding function.

4.2 Implementation

The example implicit hierarchy was built on a SGI Altix system with four 1.4 GHz Itanium 2 processors and 8 GB of main memory. The most time consuming sections of code, solving each node's sparse matrix and computing all the constraints' values, were parallelized. Computing a constraint's value is required for non-root nodes, since its value depends on the embedding functions above it in the hierarchy. The matrices were solved using the LDL solver in SGI's Scientific Computing Software Library.

5 Conclusions

We have presented a technique for generating implicit surfaces that interpolate large point sets. This method employs a hierarchical spatial partitioning that imposes a successive series of embedding functions constrained so that when they are added to one another, they interpolate the point set. The approach begins with the careful selection of a representative subset of the point set from which an interpolating implicit surface that provides a base model can be created. This base model interpolates the core subset of data points and serves as the foundation for the coarse-to-fine hierarchy. The data space is recursively divided into an octree with additional data points selected, and more detailed embedding functions are derived for each child octant that, when added to the base model, accurately interpolate the more complete, higher resolution model.

Neighboring spatial partitions are supplemented with phantom constraints that assure smooth transitions between adjoining embedding functions. No additional blending functions are required because our compactly supported radial basis functions have a limited radius of influence, imposing a predictable margin between partitions and

gradual diminishing of effect between them. Furthermore, this method elegantly handles irregularly sampled data and hole filling because of its multiresolutional approach.

A longer version of this paper with color images can be found at the following URL: `http://erie.nlm.nih.gov/hrbf`.

References

1. Savchenko, V.V., Pasko, A.A., Okunev, O.G., Kunii, T.L.: Function representation of solids reconstructed from scattered surface points and contours. Computer Graphics Forum **14**(4) (1995) 181–188
2. Turk, G., O'Brien, J.F.: Shape transformation using variational implicit functions. Computer Graphics **33**(Annual Conference Series) (1999) 335–342
3. Yngve, G., Turk, G.: Creating smooth implicit surfaces from polygonal meshes. Technical Report GIT-GVU-99-42, Georgia Institute of Technology (1999)
4. Morse, B.S., Yoo, T.S., Rheingans, P., Chen, D.T., Subramanian, K.R.: Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. In: Shape Modeling International 2001, Genoa, Italy (2001) 89–98
5. Carr, J.C., Beatson, R.K., Cherrie, J.B., Mitchell, T.J., Fright, W.R., McCallum, B.C., Evans, T.R.: Reconstruction and representation of 3D objects with radial basis functions. In: Proceedings of SIGGRAPH 2001. (2001) 67–76
6. Turk, G., O'Brien, J.F.: Modelling with implicit surfaces that interpolate. ACM Transactions on Graphics **21**(4) (2002) 855–873
7. Dinh, H., Turk, G., Slabaugh, G.: Reconstructing surfaces by volumetric regularization using radial basis functions. IEEE Trans. on Pattern Analysis and Machine Intelligence (2002)
8. Wendland, H.: Fast evaluation of radial basis functions: Methods based on partition of unity. In Chui, C.K., Schumaker, L.L., Stöckler, J., eds.: Approximation Theory X: Wavelets, Splines, and Applications, Vanderbilt University Press, Nashville, TN (2002) 472–483
9. Kojekine, N., Hagiwara, I., Savchenko, V.: Software tools using CSRBFs for processing scattered data. Computers & Graphics **27**(2) (2003) 311–319
10. Ohtake, Y., Belyaev, A., Seidel, H.: A multi-scale approach to 3d scattered data interpolation with compactly supported basis functions. In: Shape Modeling International 2003. (2003)
11. Ohtake, Y., Belyaev, A., Alexa, M., Turk, G., Seidel, H.: Multi-level partition of unity implicits. ACM TOG (Proc. SIGGRAPH 2003) **22**(3) (2003) 463–470
12. Tobor, I., Reuter, P., Schlick, C.: Multiresolution reconstruction of implicit surfaces with attributes from large unorganized point sets. In: Proceedings of Shape Modeling International (SMI 2004). (2004) 19–30
13. Ohtake, Y., Belyaev, A., Seidel, H.P.: 3d scattered data approximation with adaptive compactly supported radial basis functions. In: Shape Modeling International 2004. (2004)
14. Shen, C., O'Brien, J.F., Shewchuk, J.R.: Interpolating and approximating implicit surfaces from polygon soup. In: Proceedings of ACM SIGGRAPH 2004, ACM Press (2004) 896–904
15. Nielson, G.M.: Radial hermite operators for scattered point cloud data with normal vectors and applications to implicitizing polygon mesh surfaces for generalized CSG operations and smoothing. In: 15th IEEE Visualization 2004 (VIS'04). (2004) 203–210
16. Morse, B., Liu, W., Otis, L.: Empirical analysis of computational and accuracy tradeoffs using compactly supported radial basis functions for surface reconstruction. In: Proceedings Shape Modeling International (SMI'04). (2004) 358–361
17. Floater, M., Iske, A.: Multistep scattered data interpolation using compactly supported radial basis functions. Journal of Comp. Appl. Math. **73** (1996) 65–78
18. Iske, A., Levesley, J.: Multilevel scattered data approximation by adaptive domain decomposition. In: Numerical Algorithms. Volume 39. (2005) 187–198